Лабораторная работа №1
по дисциплине
«Методы программирования»

Сравнение алгоритмов сортировки

Работу выполнил

Саркисянц Юрий Григорьевич

Студент СКБ 221

Работу проверил

Сластников Сергей Александрович

Список файлов

Файлы

Полный список файлов.

Классы

Структура Apartment

Структура, описывающая квартиру.

Открытые члены

- bool **operator>** (const **Apartment** &other) const
  Оператор сравнения "больше" для сортировки по убыванию площади.

- bool **operator<** (const **Apartment** &other) const
  Оператор сравнения "меньше".

- bool **operator<=** (const **Apartment** &other) const
  Оператор сравнения "меньше или равно".

- bool **operator>=** (const **Apartment** &other) const
  Оператор сравнения "больше или равно".

Открытые атрибуты

- int **building_number**
  Номер дома.

- int **apartment_number**
  Номер квартиры.

- int **room_count**
  Количество комнат.

- double **total_area**
  Общая площадь квартиры.

- std::string **owner_name**
  Имя владельца.

- int **residents**
  Количество жильцов.

Подробное описание

Структура, описывающая квартиру.

Методы

bool Apartment::operator< (const Apartment & other) const

Оператор сравнения "меньше".

bool Apartment::operator<= (const Apartment & other) const

Оператор сравнения "меньше или равно".

bool Apartment::operator> (const Apartment & other) const

Оператор сравнения "больше" для сортировки по убыванию площади.

Аргументы

other	Другая квартира.

Возвращает

true, если текущая квартира должна быть раньше.

bool Apartment::operator>= (const Apartment & other) const

Оператор сравнения "больше или равно".

Данные класса

int Apartment::apartment_number

Номер квартиры.

int Apartment::building_number

Номер дома.

std::string Apartment::owner_name

Имя владельца.

int Apartment::residents

Количество жильцов.

int Apartment::room_count

Количество комнат.

double Apartment::total_area

Общая площадь квартиры.

Объявления и описания членов структуры находятся в файле:
myy.cpp

Файлы

Файл myy.cpp

```
#include <iostream>
#include <vector>
#include <fstream>
#include <sstream>
#include <chrono>
#include <algorithm>
```

Классы

struct **Apartment***Структура, описывающая квартиру.*

Функции

- template<class T> void **bubbleSort** (T a[], long size)
  Реализация пузырьковой сортировки.

- template<class T> void **insertionSort** (T a[], long size)
  Реализация сортировки вставками.

- template<class T> void **quickSortR** (T *a, long N)
  Быстрая сортировка (рекурсивная).

- std::vector< **Apartment** > **data_in** (const std::string &filename)
  Считывает данные из CSV-файла и создает вектор квартир.

- void **data_out** (const std::string &filename, **Apartment** arr[], long size)
  Записывает отсортированный массив в CSV-файл.

Функции

template<class T> void bubbleSort (T a[], long size)

Реализация пузырьковой сортировки.

Параметры шаблона

T	Тип сортируемого массива.

Аргументы

a	Массив.
size	Размер массива.

std::vector< Apartment > data_in (const std::string & filename)

Считывает данные из CSV-файла и создает вектор квартир.

Аргументы

filename	Имя файла.

Возвращает

Вектор объектов **Apartment**.

void data_out (const std::string & filename, Apartment arr[], long size)

Записывает отсортированный массив в CSV-файл.

Аргументы

filename	Имя выходного файла.
arr	Массив квартир.
size	Размер массива.

template<class T> void insertionSort (T a[], long size)

Реализация сортировки вставками.

Параметры шаблона

T	Тип сортируемого массива.

Аргументы

a	Массив.
size	Размер массива.

template<class T> void quickSortR (T * a, long N)

Быстрая сортировка (рекурсивная).

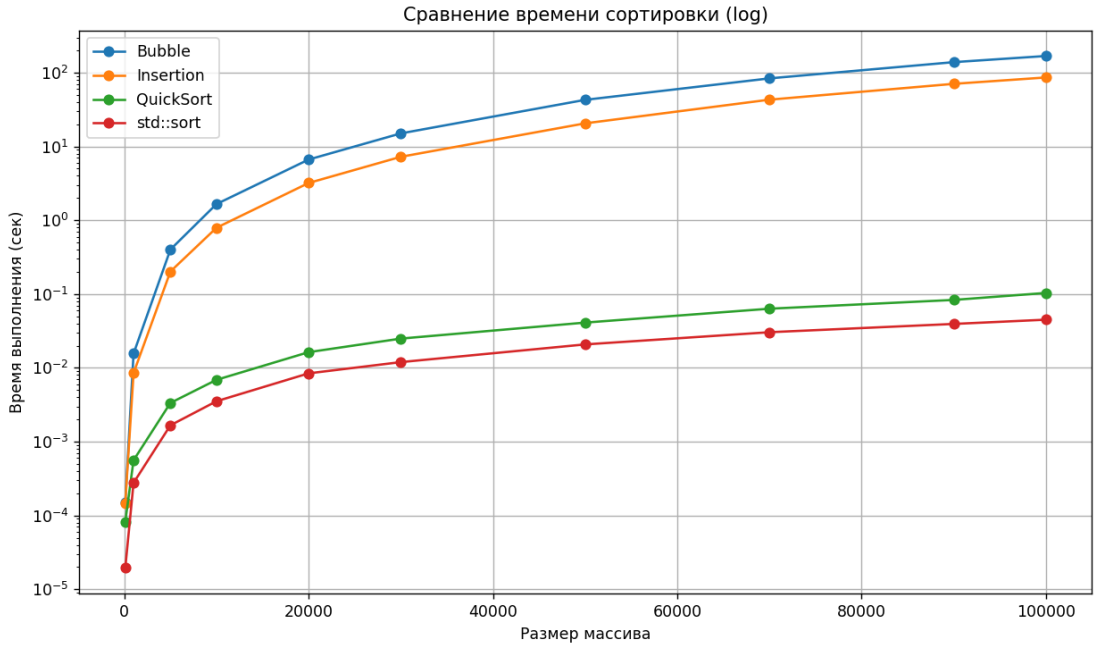
Параметры шаблона

T	Тип сортируемого массива.

Аргументы

a	Массив.
N	Размер массива.

График с сравнением времени сортировок



Исходный датасет apartments_100.txt

```
13,299,5,83.4,Сидоров Николай Сергеевич,5
16,123,3,118.6,Волкова Ирина Алексеевна,1
15,171,5,101.8,Фёдоров Алексей Сергеевич,4
8,44,5,40.3,Морозов Дмитрий Николаевич,2
11,181,2,23.4,Петрова Мария Андреевна,5
1,65,3,63.5,Морозов Алексей Алексеевич,2
6,273,1,105.6,Петрова Мария Андреевна,4
18,88,4,76.3,Иванов Алексей Сергеевич,3
3,72,4,25.0,Сидоров Сергей Андреевич,1
7,151,4,134.4,Фёдорова Светлана Дмитриевна,2
17,180,3,139.1,Новиков Андрей Сергеевич,5
9,296,4,71.1,Волкова Мария Андреевна,4
6,54,1,135.7,Соколова Татьяна Дмитриевна,2
10,254,2,116.0,Фёдоров Андрей Михайлович,5
2,91,5,109.2,Смирнов Дмитрий Андреевич,6
1,34,2,37.6,Новикова Ирина Андреевна,5
19,287,3,43.0,Морозов Алексей Сергеевич,4
15,82,5,29.1,Фёдоров Николай Андреевич,5
20,41,2,85.2,Кузнецов Николай Алексеевич,4
1,200,5,33.3,Соколова Светлана Петровна,3
18,246,2,116.3,Петрова Ирина Анатольевна,4
10,76,3,65.8,Петров Николай Иванович,5
8,81,2,121.5,Васильев Иван Владимирович,4
19,248,3,106.3,Иванова Ольга Анатольевна,5
10,98,4,103.3,Фёдоров Сергей Иванович,3
4,104,3,56.1,Волкова Наталья Михайловна,4
7,203,5,76.2,Иванов Дмитрий Анатольевич,5
14,201,1,80.9,Фёдорова Светлана Владимировна,6
2,168,1,45.3,Сидорова Татьяна Дмитриевна,2
6,13,5,47.8,Иванов Владимир Сергеевич,1
15,158,4,20.8,Сидоров Алексей Иванович,3
7,121,3,72.6,Петрова Ирина Дмитриевна,5
```

2,27,2,63.5,Новикова Мария Владимировна,1
7,58,4,21.1,Смирнов Владимир Николаевич,3
2,102,4,21.8,Новиков Иван Иванович,6
20,151,5,77.3,Кузнецова Мария Анатольевна,2
4,217,3,54.6,Соколов Дмитрий Сергеевич,4
20,208,4,64.1,Сидоров Николай Анатольевич,4
16,21,1,76.8,Новикова Ирина Михайловна,2
13,181,1,40.4,Новикова Ольга Анатольевна,5
1,190,2,73.4,Иванова Анна Владимировна,5
4,155,3,94.0,Петрова Татьяна Владимировна,3
15,104,3,98.6,Новикова Ольга Дмитриевна,3
11,53,3,120.0,Иванов Алексей Иванович,6
7,185,5,122.0,Волков Алексей Андреевич,4
18,245,3,72.2,Сидоров Андрей Анатольевич,4
9,255,5,107.9,Иванов Иван Владимирович,3
8,211,5,78.8,Попов Николай Анатольевич,6
7,39,4,95.6,Иванова Ирина Владимировна,2
6,155,2,142.3,Фёдоров Михаил Николаевич,5
1,18,4,130.9,Соколов Андрей Алексеевич,4
9,32,3,105.7,Соколов Андрей Андреевич,3
18,172,3,119.9,Попов Сергей Михайлович,1
18,287,2,140.0,Попов Дмитрий Иванович,2
14,170,3,46.6,Попова Елена Анатольевна,5
9,153,3,98.5,Кузнецова Ольга Петровна,5
7,42,5,134.7,Васильев Иван Сергеевич,5
3,104,5,22.3,Петрова Анна Алексеевна,1
19,276,5,55.7,Кузнецова Татьяна Анатольевна,4
5,279,1,147.7,Волкова Анна Сергеевна,2
20,155,4,94.7,Волкова Анна Петровна,3
15,25,5,40.9,Соколов Дмитрий Дмитриевич,4
15,168,4,140.2,Петрова Елена Сергеевна,2
14,56,4,48.2,Новиков Михаил Сергеевич,5
10,84,3,146.1,Смирнов Николай Дмитриевич,5
4,205,5,26.2,Кузнецов Иван Владимирович,1
14,104,5,46.1,Соколов Дмитрий Николаевич,6
7,297,4,115.6,Попова Ирина Анатольевна,5
10,193,2,141.2,Морозов Сергей Иванович,5
2,221,4,79.7,Петрова Татьяна Сергеевна,6
6,181,3,77.8,Васильев Алексей Николаевич,6
12,166,4,125.4,Смирнов Алексей Андреевич,5
16,84,5,57.7,Иванова Анна Анатольевна,1
1,300,5,66.0,Васильева Мария Владимировна,4
13,274,5,90.2,Морозов Иван Сергеевич,3
11,163,2,117.0,Васильева Татьяна Владимировна,4
15,131,5,68.7,Сидоров Михаил Владимирович,2
11,198,1,29.7,Иванов Иван Николаевич,5
7,164,3,99.8,Соколов Николай Дмитриевич,1
2,173,2,133.4,Волков Сергей Алексеевич,1
11,187,3,78.7,Соколов Сергей Сергеевич,5
1,242,3,63.3,Васильева Елена Николаевна,5
2,35,4,85.4,Волкова Мария Михайловна,4
14,265,4,21.3,Смирнова Ирина Алексеевна,4
10,184,4,129.3,Сидорова Анна Алексеевна,2
6,40,2,146.4,Фёдорова Светлана Николаевна,6
5,227,1,144.9,Соколова Наталья Дмитриевна,6
18,259,4,145.5,Морозова Ольга Андреевна,5
8,195,5,48.6,Петрова Анна Ивановна,6
2,52,4,42.2,Иванова Ольга Андреевна,3
18,197,1,87.5,Волкова Ольга Ивановна,4
18,58,3,73.1,Васильев Иван Михайлович,2
15,225,3,51.9,Васильева Мария Ивановна,5

```
4,232,4,147.3,Волкова Ольга Алексеевна,4
4,56,5,49.4,Соколов Андрей Иванович,6
8,190,1,38.5,Волкова Елена Анатольевна,5
4,145,5,138.3,Морозов Дмитрий Николаевич,3
13,154,4,36.0,Смирнов Алексей Михайлович,3
3,21,2,94.6,Соколова Светлана Андреевна,5
5,190,5,134.1,Соколов Владимир Владимирович,3
```

Полный файл my.cpp

```
#include <iostream>
#include <vector>
#include <fstream>
#include <sstream>
#include <chrono>
#include <algorithm>

struct Apartment {
    int building_number;
    int apartment_number;
    int room_count;
    double total_area;
    std::string owner_name;
    int residents;

    bool operator>(const Apartment& other) const {
        if (total_area != other.total_area)
            return total_area < other.total_area; // по убыванию
        if (building_number != other.building_number)
            return building_number > other.building_number;
        if (apartment_number != other.apartment_number)
            return apartment_number > other.apartment_number;
        return owner_name > other.owner_name;
    }

    bool operator<(const Apartment& other) const {
        return other > *this;
    }

    bool operator<=(const Apartment& other) const {
        return !(*this > other);
    }

    bool operator>=(const Apartment& other) const {
        return !(*this < other);
    }
};
// Сортировка пузырьком
template<class T>
void bubbleSort(T a[], long size) {
    for (long i = 0; i < size - 1; ++i) {
        for (long j = 0; j < size - i - 1; ++j) {
            if (a[j] > a[j + 1]) {
                std::swap(a[j], a[j + 1]);
            }
        }
    }
}

// Сортировка вставками
template<class T>
void insertionSort(T a[], long size) {
    for (long i = 1; i < size; i++) {
        T key = a[i];
        long j = i - 1;
        while (j >= 0 && a[j] > key) {
            a[j + 1] = a[j];
```

```
            j--;
        }
        a[j + 1] = key;
    }
}

template<class T> void quickSortR(T* a, long N) {
    long i = 0, j = N - 1;
    T p = a[N / 2], x;
    do {
        while (a[i] < p) i++;
        while (a[j] > p) j--;
        if (i <= j) {
            x = a[i]; a[i] = a[j]; a[j] = x;
            i++; j--;
        }
    } while (i <= j);

    if (j > 0) quickSortR(a, j + 1);
    if (N > i) quickSortR(a + i, N - i);
}

std::vector<Apartment> data_in(const std::string& filename) {
    std::vector<Apartment> data;
    std::ifstream inFile(filename);
    if (!inFile) {
        std::cerr << "Ошибка открытия файла: " << filename << std::endl;
        return data;
    }

    std::string line;
    while (std::getline(inFile, line)) {
        std::stringstream ss(line);
        Apartment apt;
        std::string field;

        std::getline(ss, field, ','); apt.building_number =
std::stoi(field);
        std::getline(ss, field, ','); apt.apartment_number =
std::stoi(field);
        std::getline(ss, field, ','); apt.room_count =
std::stoi(field);
        std::getline(ss, field, ','); apt.total_area =
std::stod(field);
        std::getline(ss, apt.owner_name, ',');
        std::getline(ss, field); apt.residents = std::stoi(field);

        data.push_back(apt);
    }
    return data;
}

void data_out(const std::string& filename, Apartment arr[], long size)
{
    std::ofstream outFile(filename);
    for (long i = 0; i < size; i++) {
        outFile << arr[i].building_number << ","
                << arr[i].apartment_number << ","
                << arr[i].room_count << ","
                << arr[i].total_area << ","
                << arr[i].owner_name << ","
```

```
                << arr[i].residents << "\n";
    }
    outFile.close();
}

int main() {
    std::string filenames[] = {
        "apartments_100.txt", "apartments_1000.txt",
"apartments_5000.txt",
        "apartments_10000.txt", "apartments_20000.txt",
"apartments_30000.txt",
        "apartments_50000.txt", "apartments_70000.txt",
"apartments_90000.txt",
        "apartments_100000.txt",
    };

    for (const auto& fname : filenames) {
        std::vector<Apartment> vec = data_in("in_data/" + fname);
        if (vec.empty()) continue;

        long size = vec.size();
        Apartment* baseArray = new Apartment[size];
        for (long i = 0; i < size; i++) baseArray[i] = vec[i];

        // Пузырьковая

        Apartment* bubbleArr = new Apartment[size];
        std::copy(baseArray, baseArray + size, bubbleArr);
        auto start = std::chrono::high_resolution_clock::now();
        bubbleSort(bubbleArr, size);
        auto end = std::chrono::high_resolution_clock::now();
        std::cout << fname << " | Bubble: " <<
std::chrono::duration<double>(end - start).count() << " сек\n";
        data_out("out_data/Sorted_bubble_" + fname, bubbleArr, size);
        delete[] bubbleArr;

        // Вставки
        Apartment* insertArr = new Apartment[size];
        std::copy(baseArray, baseArray + size, insertArr);
        start = std::chrono::high_resolution_clock::now();
        insertionSort(insertArr, size);
        end = std::chrono::high_resolution_clock::now();
        std::cout << fname << " | Insertion: " <<
std::chrono::duration<double>(end - start).count() << " сек\n";
        data_out("out_data/Sorted_insertion_" + fname, insertArr,
size);
        delete[] insertArr;

        // Быстрая
        Apartment* quickArr = new Apartment[size];
        std::copy(baseArray, baseArray + size, quickArr);
        start = std::chrono::high_resolution_clock::now();
        quickSortR(quickArr, size);
        end = std::chrono::high_resolution_clock::now();
        std::cout << fname << " | QuickSort: " <<
std::chrono::duration<double>(end - start).count() << " сек\n";
        data_out("out_data/Sorted_quick_" + fname, quickArr, size);
        delete[] quickArr;

        // std::sort
        Apartment* stdArr = new Apartment[size];
```

```
        std::copy(baseArray, baseArray + size, stdArr);
        start = std::chrono::high_resolution_clock::now();
        std::sort(stdArr, stdArr + size);
        end = std::chrono::high_resolution_clock::now();
        std::cout << fname << " | std::sort: " <<
std::chrono::duration<double>(end - start).count() << " сек\n";
        delete[] stdArr;

        delete[] baseArray;
    }

    return 0;
}
```